

# 1 NETWORK DATA PACKET CLASSIFICATION AND DEMULTIPLEXING

## 2 FIELD OF THE INVENTION

3 The present invention is directed to the field of packet  
4 communication. It is more particularly directed to  
5 classification and demultiplexing of network communication  
6 packets processed in a network protocol stack.

## 7 BACKGROUND OF THE INVENTION

8 Communication over a network often requires the information that  
9 is to be transported from one computer to another be divided into  
10 network communication packets. These network communication  
11 packets, simply referred to as "packets", are transported across  
12 the physical communication network.

13 The information originating from an application program becomes  
14 packetized into network communication packets by passing through  
15 various software components before arriving at the network  
16 interface card for transmission on the physical communications  
17 network. These software components are typically layered to form  
18 what is known as the network protocol stack. Each layer is  
19 responsible for a different facet of communication. For example,  
20 the TCP/IP protocol stack is normally split into four layers:  
21 link, network, transport and application. Figure 1 shows the  
22 relationship between the protocol layers and the TCP/IP protocol  
23 stack. The link layer 101 is responsible for placing data on the

1 physical network. The network layer 102 is responsible for  
2 routing. The transport layer 103 is responsible for the  
3 communication between two hosts. The application layer 104 is  
4 responsible for processing the application specific data.

5 For example, Figure 2 illustrates the stages of an HTTP request  
6 being encapsulated before being sent to a web server. As the  
7 request descends the protocol stack, each layer 201-204  
8 encapsulates the packet adding its own header. When the HTTP  
9 packet arrives at the destination address, each protocol layer  
10 uses information within its header to classify the incoming  
11 packet amongst all the protocols in the layer above it. This  
12 process is commonly referred to as demultiplexing.

13 At each layer in the network protocol stack, the packet is  
14 demultiplexed or "classified" based on information about the  
15 packet that is contained in the headers or from information  
16 inside the data portion of the packet itself. The packet is  
17 processed differently based on its classification.

18 For example, Figure 3 illustrates how this classification is done  
19 for an incoming HTTP request 301. The Ethernet driver 302, in  
20 the link layer 300, classifies the packet based on frame type in  
21 the Ethernet header and passes it to IPv4 312 in the network  
22 layer 310. IPv4 312 classifies the packet based on the IP header  
23 protocol value in the IP header and passes it to TCP 323 in the  
24 transport layer 320. TCP classifies the packet based on the  
25 destination port number in the TCP header and passes it to the  
26 HTTP server 332 in the application layer 330.

1 Traditional packet classification systems, as found in BPF, DPF,  
2 Pathfinder, Router Plugins, operating systems and many firewalls,  
3 are limited to a set of fixed pattern matching rules. This  
4 allows a user to intercept/process any packet that matches the  
5 desired set of values in the appropriate byte ranges (usually a  
6 combination of the IP and the protocol header fields, such as  
7 source/destination address, protocol or source/destination  
8 ports). These packets are then passed to a software module that  
9 processes the packets and can modify, forward, drop or delay  
10 them. Stateful packet filtering systems generally have the  
11 ability to generate and add rules dynamically based on  
12 application traffic. However, such systems do not provide simple  
13 methods to extend packet processing to understand new application  
14 protocols.

15 These traditional systems may work well for applications that use  
16 a single connection to a well known destination address and port.  
17 However, many modern applications initially use a well known  
18 service port for the control session and then use additional  
19 connections on ephemeral port numbers for each data stream.  
20 Examples of such applications are FTP, Real Audio and H.323. To  
21 support these applications efficiently, the traditional systems  
22 must allow packet matching filter rules to be updated dynamically  
23 and quickly. In addition, some modern protocols have abandoned  
24 using fixed format headers and fixed sized fields. For example,  
25 HTTP makes its header human readable by encoding them as  
26 strings.

## 27 SUMMARY OF THE INVENTION

1 It is thus an aspect of the present invention to provide greater  
2 flexibility in classifying and demultiplexing packets in the  
3 network protocol stack. As a result, it provides a method for  
4 application level classification. This is due to classifying  
5 techniques and a modular structure described subsequently.

6 Another aspect of the present invention provides easier  
7 extendibility for packet processing in the network protocol stack  
8 by defining a standard method for adding new functionality or  
9 support for new protocols and applications.

10 Another aspect of the present invention provides methods and  
11 apparatus to obtain external information, from an application  
12 scheduled outside of the forwarding or interrupt context of the  
13 kernel, in order to augment packet classification and/or  
14 disposition.

15 An example embodiment of the present invention is a method for  
16 classifying a data packet. The method includes the steps of:  
17 receiving the packet at a root node of a classification tree;  
18 passing the packet to a first child node of a first tree level of  
19 the classification tree indicating a satisfaction of a  
20 node-criteria of the first child node; the first child node  
21 forming the data packet into a matched packet; and repeating the  
22 step of passing and forming for a next tree level until no first  
23 child node of the next level at a succeeding next level indicates  
24 satisfaction of the node-criteria of the first child node of the  
25 next level.

1 In some embodiments the step of indicating includes the step of  
2 executing a set of code which returns a status indication of the  
3 type; and/or the step of indicating satisfaction of a criteria  
4 includes the steps of executing a set of code which identifies  
5 the desired packet and returning a status indication; and/or the  
6 step of forming the data packet into a matched packet includes  
7 the step of indicating satisfaction; and/or the step of repeating  
8 the step of passing and the step forming includes the steps of  
9 indicating and returning a status indication of NO\_Match.

10 In some embodiments of the method, the method further includes:  
11 the step of adding at least one new child node; and/or one new  
12 child node is a Real Audio node; and/or the method is extendible  
13 such that one or more nodes are dynamically added at any level;  
14 parsing the matched packet and generating relevant information;  
15 transforming the matched packet into a transformed packet; and/or  
16 associating the packet at a last first child node indicating  
17 satisfaction; executing a set of code in accordance with the last  
18 first child node; and/or the step of forming includes the first  
19 child node specifying a set of code to be run subsequently;  
20 and/or the step specifying specifies the set of code to be run  
21 following classification.

22 Another example embodiment of the present invention is a method  
23 which uses an external process for classifying a packet. This  
24 method includes the steps of suspending a classification process  
25 in progress for the packet, and obtaining external information  
26 employed in the classifying. This is performed by an  
27 application scheduled outside of the forwarding or interrupt  
28 context of the kernel.

1 In some embodiments of the method, the step of suspending  
2 includes the steps of queuing any data, including information  
3 about the packet or its present classification; and/or  
4 transferring said data to an application that is scheduled  
5 outside of the forwarding or interrupt context of the kernel.

6 In some embodiments of the method, the step of obtaining external  
7 information includes augmenting a node-criteria of a node in a  
8 classification tree with additional information; and/or the  
9 external information includes authentication of an originator of  
10 the packet; the classification process is an extendible  
11 classifier process (In one application, a process is extendible  
12 by adding a new child node); and/or the step of specifying  
13 includes enforcement of a site policy. A site policy is  
14 composed of a number of different aspects including security.  
15 The security aspect of a site policy may be based on packet  
16 classification and authentication information.

17 Another aspect of the present invention is a method for  
18 determining disposition of an original packet received at a child  
19 node. The method includes the step of passing the original  
20 packet and a first disposition of the original packet to an  
21 external process, and the external process augmenting the  
22 original packet and/or augmenting the first disposition by  
23 employing a process specific means and returning an augmented  
24 packet and an augmented disposition to the child node. Some  
25 embodiments of the method include suspending a disposition  
26 process in progress for the original packet; and/or the augmented

1 disposition includes identification of and/or authentication of  
2 an originator of said packet.

3 **BRIEF DESCRIPTION OF THE DRAWINGS**

4 These and other aspects, features, and advantages of the present  
5 invention will become apparent upon further consideration of the  
6 following detailed description of the invention when read in  
7 conjunction with the drawing figures, in which:

8 Fig. 1 shows the relationship between the protocol layers and the  
9 TCP/IP protocol stack;

10 Fig. 2 illustrates the stages of an HTTP request being  
11 encapsulated before being sent to a web server;

12 Fig. 3 illustrates how classifying is done for an incoming HTTP  
13 request;

14 Fig. 4 shows an example of how to organize modules in the  
15 classification tree in accordance with the present invention;

16 Fig. 5 shows an example of a packet classification and  
17 demultiplexing process in classifying a packet in accordance with  
18 the present invention;

19 Fig. 6 shows an example of steps to determine the packet  
20 disposition in accordance with the present invention;

1 Fig. 7 shows an example of pm\_t return codes in accordance with  
2 the present invention;

3 Fig. 8 shows an example of application dependent nodes in  
4 accordance with the present invention;

5 Fig. 9 shows an example of pp\_t return codes in accordance with  
6 the present invention;

7 Fig. 10 shows an example of paction\_t return codes in accordance  
8 with the present invention; and

9 Fig. 11 shows an example of an apparatus in accordance with the  
10 present invention.

#### 11 DETAILED DESCRIPTION OF THE INVENTION

12 Networking protocols are normally divided into layers which are  
13 responsible for different facets of communication as Figure 1  
14 depicts for the network layers of the TCP/IP protocol. The  
15 associated call graph created by the standard UNIX protocol stack  
16 is arranged like a tree as described for Figure 3. Each level of  
17 the tree corresponds to a different layer in the networking  
18 protocol stack. The present invention mimics the call graph of  
19 the UNIX protocol stack and organizes the different modules  
20 competing for packets at the IP layer in a tree structure herein  
21 referred to as a classification tree.



1 An example of a classification tree 400 is shown in Figure 4.  
2 Figure 4 shows each node in the classification tree as a separate  
3 module. In an embodiment of the present invention each node is  
4 composed of 4 packet traversal functions (matcher, preprocessor,  
5 action, and post processing) and 3 node management functions  
6 (callback, heartbeat and management). Only the packet matching  
7 function, which identifies the packets to process, and the packet  
8 action function, which determines the packet disposition, are  
9 required. The packet matching function is herein referred to as  
10 the node-criteria of the node. The remaining traversal and  
11 management function pointers can default to NULL. These  
12 functions associated with each node are stored in a PacketFilter  
13 structure.

14 Since each of the nodes is a separate dynamically loadable  
15 module, the classification tree organization is flexible. In an  
16 embodiment of the present invention, the modules are loaded into  
17 memory during the initialization process. Based upon  
18 configuration information the modules are then arranged to form a  
19 classification tree. The ordering of the modules is important  
20 since the packet traversal is governed by this ordering. As the  
21 classification tree is created, each node is initialized by  
22 executing a set of code. In the embodiment, this set of code is  
23 a function referred to as the management function(*mm*). The input  
24 parameter to the *mm* function is generally a single pointer to a  
25 buffer containing the node specific configuration data.

26 Figure 4 shows an example of how to organize modules in the  
27 classification tree. The IPv4 503, IPv6 504, UDP 506, HTTP 507  
28 and TCP 508 modules each wish to observe or modify packets that

1 use the protocol for which they are named. However, in this  
2 example, there are multiple ways one could imagine wanting to  
3 process HTTP requests. These ways include: providing a  
4 transparent HTTP proxy function, using a specialized TCP for HTTP  
5 like Transaction TCP(T/TCP), doing content filtering based on  
6 site policy, or limiting packet traffic based on a service  
7 contract. Depending on the intended purpose of the  
8 classification tree, differing modules are loaded into memory. A  
9 site policy is composed of a number of different aspects  
10 including security. The security aspect of a site policy may be  
11 based on packet classification and authentication information.  
12 Once initialization completes, the classification tree may be  
13 modified by adding, deleting or moving a node. This ability of  
14 modifying the classification tree makes the packet classification  
15 process extendible.

16 The present invention includes methods for implementing a packet  
17 classification process and/or an augmented packet disposition  
18 process. The packet to be classified and/or augmented is herein  
19 referred to as the original packet. The resulting packet is  
20 referred to as the augmented packet. The disposition of the  
21 original packet is herein referred to as the first disposition,  
22 and the disposition resulting from the augmented disposition  
23 process is herein referred to as the augmented disposition.  
24 Anything outside of the forwarding or interrupt context of the  
25 kernel is herein said to be external.

26 An example embodiment has 7 steps to classify a packet and  
27 determine the augmented packet disposition. These steps are in  
28 the interrupt context except where noted. Steps 1-4 describe the

1 packet classification process shown in Figure 5. Steps 5-7  
2 describe augmenting the packet disposition process. A flow  
3 diagram for these seven steps is shown in Figure 6. Refer to  
4 figures 5 & 6 for the following description.

5 Step 1: After receiving a packet from the physical network, the  
6 Link Layer passes the packet to the root node 502.

7  
8 In this step, a network driver receives a packet from the  
9 physical network, which it classifies based on frame type in  
10 the MAC header and passes it to the root node of the  
11 classification tree.

12  
13 Step 2: The packet is passed to a first child node of the first  
14 level 521 of the classification tree, indicating a satisfaction  
15 of a node-criteria of the child node.

16 The root node asks each child node from left to right whether  
17 the packet matches its node-criteria, until a child node's  
18 node-criteria is satisfied. The root node then passes the  
19 packet to that first child node which satisfies the  
20 node-criteria and forms the data packet into a matched packet.  
21 In Figure 5, the root node 502 first passes the packet to the  
22 IPv4 node 503. A child node's node-criteria includes a set  
23 of code used to identify the packets desired. This set of  
24 code is implemented as a function referred to as the packet  
25 matching function (*pm*) 603.

26 The input parameters to the *pm* function are: the PBUF, an  
27 operating system independent data structure containing the

1 packet, the options memory area, and a pointer to the packet  
2 filter node. The result of the packet matching function,  
3 indicating satisfaction or lack thereof, of the child node's  
4 node-criteria, is of type *pm\_t*. Figure 7 enumerates a sample  
5 group of type *pm\_t* return code values 700. The packet  
6 matching function results indicating satisfaction of a child  
7 node's node-criteria include: Match\_OK, Match\_This,  
8 Match\_Discard, and Match\_Forward. The result indicating lack  
9 of satisfaction is NO\_Match.

10 The packet matching function may be as simplistic as matching  
11 a static fixed offset, such as the IPv4 node, or as complex as  
12 identifying the packets for applications which negotiate  
13 additional connections, such as FTP, Real-Audio and H.323.  
14 Unfortunately, since each of these applications has its own  
15 method for negotiating additional connections, application  
16 dependent nodes are required. This is as illustrated in  
17 Figure 8 for H.323 831, Real-Audio 832, and FTP 833. For each  
18 additional connection, a dynamic filter rule is created. These  
19 dynamic filter rules and other state information for the  
20 negotiated connections are stored locally in the application  
21 specific node. One implementation uses a hashtable structure  
22 for storing this data. Based on the well known services port  
23 and the application specific data, the packet matching  
24 function identifies the packets desired enabling application  
25 level classification.

26 Step 3: Repeat the process of 'passing the packet' starting with  
27 a first child node of a next tree level of the classification  
28 tree which satisfies a node-criteria of that first child node, as

1 described in step 2, and form the packet into a matched packet,  
2 until no child of a next tree level of the classification tree  
3 succeeds in satisfying a node-criteria (No\_Match).

4 A determination is made if there is a next child 604. If  
5 there is, flow continues with 601. If not, flow continues  
6 with 621. Thus, when the packet matching function of all of  
7 the children nodes of the next tree layer result in a lack of  
8 satisfaction, (No\_Match), the packet is said to have fully  
9 traversed the classification tree. The traversal path is  
10 defined as the set of nodes from the root to the last first  
11 child node satisfying a node-criteria of the child node. Thus  
12 packet classification has completed and flow continues with  
13 621.

14 Step 4: For each first child node, satisfying a node-criteria of  
15 the child node form the data packet into a matched packet. This  
16 may be performed as in steps 4A, 4B and/or 4C.

17 Step 4A: The current node is added to the node  
18 traversal path 605.

19 Step 4B: The node may execute a set of code, if such a  
20 code exists, which may parse and transform the  
21 packet 607. If the set of code exists, the packet  
22 is parsed and transformed 617. If not, flow  
23 continues with 609.

24 Once a node's node-criteria is satisfied, the packet's  
25 traversal of the classification tree is limited to the  
26  
27

node's descendants. The remainder of the classification tree is not traversed. But before traversing a node's subtree, the node may execute a set of code. In the present embodiment, this set of code is referred to as the packet preprocessor function(*pp*). The input parameters are the same as the packet matching function. This includes: the PBUF, an operating system independent data structure containing the packet, the options memory area, and a pointer to the packet filter node. The return code of the *pp* function is of type *pp\_t*. Examples of type *pp\_t* 900 are enumerated in Figure 9. The packet preprocessor function may perform actions such as parsing a packet and transforming a packet. Parsing a packet generates information that may need to be made available to the node's descendants and ancestors. Transforming a packet takes place for example, when the IPSEC node's preprocessor transforms an encrypted packet into a decrypted packet. IPsec tunnel information and other information is created that can be used by other nodes in the classification tree.

The present invention thus provides a generic mechanism for retaining or passing state information between nodes by a mechanism we referred to herein as options passing. In an example of option passing, an options memory segment is attached to each packet during its tree traversal. Each node may store and retrieve state using the APIs: *fw\_add\_option*, *fw\_next\_option*. Since a node may not understand all options that are passed to it, the

node will process the options it understands and ignore those which it does not understand.

Step 4C: The node may also suspend the classification process in order to obtain additional external information so as to augment packet classification and demultiplexing 615.

Suspending the classification process involves queueing any data, including information about the packet or its present classification, and transferring the data to an application that is scheduled outside of the forwarding or interrupt context of the kernel.

One embodiment augments packet classification by suspending the packet classification process until the application, scheduled outside of the forwarding or interrupt context of the kernel, completes. The resulting external information is used to augment the packet classification.

Examples of applications which may augment packet classification include packet identification and authentication agents. An identification/authentication agent, may use s/ident for out of band identification and authentication. Authentication may use s/ident for out of band authentication in order to correlate the packet with a userid. Another example of authentication is to correlate a VPN tunnel id with a userid.

1           This external information, such as packet  
2           identification and/or authentication, permits packets to  
3           be handled differently. For example, assume that a site  
4           connected to the Internet is severely bandwidth limited.  
5           As a result only a limited number of employees at any  
6           given moment can run applications with high bandwidth  
7           demands, such as streaming media. Based on the external  
8           information a site policy can be implemented which gives  
9           preferential treatment to a set of employees.

10   Step 5: After packet classification completes, a set of code  
11   associated with the last child node which satisfied the  
12   node-criteria, is executed.

13  
14   In an embodiment, this set of code is referred to as the  
15   packet action function(*pa*). The packet action input  
16   parameters are: the PBUF, a pointer to the node, a pointer to  
17   the node traversal path, and the options memory area. An  
18   example of return codes of the *pa* function, of type *paction\_t*  
19   are enumerated in Figure 10 1000. The return code obtained  
20   determines the packet disposition.

21   Normally the packet action function 621 monitors the packet  
22   data in order to obtain application specific state information  
23   used by the other node functions. For example, a packet  
24   action function with application specific knowledge could  
25   monitor the packet data for new negotiated data connections.  
26   These new dynamic connections are stored locally in the  
27   application specific node. The packet matching function uses



1 the dynamic data as part of the node-criteria for application  
2 level packet classification.

3 Other examples of packet action function usage include:  
4 modifying packets, which may be used to implement NAT;  
5 queueing packets, which may be used to shape traffic; dropping  
6 packets, which may be used for rate limiting; and redirecting  
7 packets, which may be used for load balancing.

8  
9 The packet action function may also suspend kernel packet  
10 processing and transfer any data (including information about  
11 the packet or its classification) to an application that is  
12 scheduled outside of the forwarding or interrupt context of  
13 the kernel, and/or to obtain external information in order to  
14 augment the packet disposition (i.e. discard, forward, process  
15 locally or redirect) decision 631. Suspending the packet  
16 disposition decision process involves queueing any data,  
17 including information about the packet or its classification,  
18 and transferring the data to an application employing a  
19 process specific means that is scheduled outside of the  
20 forwarding or interrupt context of the kernel.

21 An example method of augmenting the packet disposition  
22 decision is to suspend any in progress packet disposition  
23 decision process until the application scheduled outside of  
24 the forwarding or interrupt context of the kernel completes.  
25 The resulting external information is used to augment the  
26 packet disposition decision. Examples of applications which  
27 may augment the packet disposition decision are policy  
28 enforcement and content filtering agents based on any

1 combination of the packet classification, identification and  
2 authentication. Examples of process specific means include  
3 s/identd and external LDAP servers.

4 Once the application completes, it passes the original data,  
5 the external information and the results to the kernel, which  
6 issues a call to the node's callback function. The callback  
7 function(*cb*) reinserts the packet at the node which suspended  
8 the processing. Based on the application's results, dynamic  
9 rules may be created 621.

10 For example, an especially advantageous usage is with VPN  
11 tunnels. Differing policies based on the VPN callee are  
12 enforceable using dynamic rules. With application level  
13 classification, these rules are no longer limited to fixed  
14 pattern matches, such as protocol, but may be written in terms  
15 of applications. An example of an application level rule  
16 would be 'permit John Doe Real-Audio'. Application level  
17 rules would also simplify firewall rule definitions in  
18 firewall applications.

19 Step 6: After the set of code associated with the last child  
20 node, which satisfied the node-criteria (referred to as the  
21 packet action code) completes, a set of code associated with each  
22 node in the node traversal path, is executed 623.

23 In the present embodiment, this set of code is referred to as  
24 the packet post processor function(*px*) 625. The packet  
25 postprocessing input parameters are the PBUF, the options  
26 memory area and the packet action disposition. The return

code of the *px* function is of type *paction\_t*. Examples of type *paction\_t* are enumerated in Figure 10.

Just as the packet preprocessing may decrypt a packet, the packet postprocessing may perform actions such as encrypting a packet 627. As the packet originally traversed the classification tree, the node traversal path was created. Before returning to the base operating system, in reverse node traversal order, packet postprocessing is executed.

Normally, the packet disposition is maintained through postprocessing. Only in unusual circumstances does the postprocessing not follow the recommended packet action and previous post processing disposition. For example, with VPN tunnels the outbound tunnel may have been torn down during the classification tree traversal.

Step 7: After the packet processing completes, control returns to the base operating system, which discards, forwards, redirects or locally processes the packet, based on the final disposition 633.

Figure 11 shows an example embodiment of the present invention as an apparatus to classify and/or augment the disposition of a data packet shown in Figure 11. The apparatus includes a network interface device (1101) to receive a packet from the physical network and, pass the packet to a root node of a classification tree, and the reverse, to receive a packet from the root node and send a packet to the physical network. The apparatus also includes a packet module (1103) to successively pass the packet from child node to child node on each tree level until a first

1 child node of a tree level of the classification tree indicates a  
2 satisfaction of a node-criteria of that first child node. The  
3 first child node forms the data packet into a matched packet  
4 until no first child node of a next level at a succeeding next  
5 level indicates satisfaction of the node-criteria of the first  
6 child node of the next level.

7 It is noted that an accelerator chip can be used to implement the  
8 packet module (1103). This chip can be used as the basis of a  
9 firewall box, a border server, or as an application level  
10 classification system such as needed when diagnosing high speed  
11 networking problems.

12 Other apparatus embodiments of the present invention may be  
13 implemented in ways known to those familiar with the art. For  
14 example, the invention may be implemented using an apparatus for  
15 classifying a data packet. This apparatus includes: means for  
16 receiving the data packet at a root node of a classification  
17 tree; means for successively passing the data packet to each  
18 child of a first tree level until a first child node of the first  
19 tree level of the classification tree indicates a satisfaction of  
20 a node-criteria of said first child node, and the first child  
21 node forming said data packet into a matched packet; and means  
22 for repeating the steps of passing and forming for a next tree  
23 level until no first child node of said next tree level at a  
24 succeeding next level indicates satisfaction of the node-criteria  
25 of said first child node of said succeeding next level. This  
26 apparatus may, for example, be in the form of a floppy or hard  
27 disk, flash memory, or external magnetic media, etc.

1 Another example embodiment of the present invention is an  
2 apparatus for determining disposition of a packet received at a  
3 child node. This apparatus includes: an interrupt context of a  
4 control program, with the child node existing within the  
5 interrupt context; an external process outside of the interrupt  
6 context of the control program; means for passing said packet and  
7 a first disposition of said packet to the external process, the  
8 external process to augment the packet disposition by employing a  
9 process specific means and to return an augmented packet with an  
10 augmented disposition to the child node; and the interrupt  
11 context including means for receiving the augmented packet and  
12 the augmented disposition from the external process. This  
13 apparatus may, for example, also be in the form of a hard disk, a  
14 floppy disk, or external magnetic media, etc. A control program  
15 may be implemented as software that manages the example  
16 apparatus.

17 The present invention can be realized in hardware, software, or a  
18 combination of hardware and software. The present invention can  
19 be realized in a centralized fashion in one computer system, or  
20 in a distributed fashion where different elements are spread  
21 across several interconnected computer systems. Any kind of  
22 computer system - or other apparatus adapted for carrying out the  
23 methods described herein - is suited. A typical combination of  
24 hardware and software could be a general purpose computer system  
25 with a computer program that, when being loaded and executed,  
26 controls the computer system such that it carries out the methods  
27 described herein. The present invention can also be embedded in  
28 a computer program product, which comprises all the features  
29 enabling the implementation of the methods described herein, and

1 which - when loaded in a computer system - is able to carry out,  
2 or cause the carrying out of these methods.

3 Computer program means or computer program in the present context  
4 mean any expression, in any language, code or notation, of a set  
5 of instructions intended to cause a system having an information  
6 processing capability to perform a particular function either  
7 directly or after either or both of the following:

- 8 1. conversion to another language, code or notation; and/or
- 9 2. reproduction in a different material form.

10 It is noted that the foregoing has outlined some of the more  
11 pertinent objects and embodiments of the present invention. The  
12 concepts of this invention may be used for many applications.  
13 Thus, although the description is made for particular  
14 arrangements and methods, the intent and concept of the invention  
15 is suitable and applicable to other arrangements and  
16 applications. For example, although reference is made to a data  
17 packet, the invention is similarly applicable to a non-data  
18 packet. It will be clear to those skilled in the art that other  
19 modifications to the disclosed embodiments can be effected  
20 without departing from the spirit and scope of the invention.  
21 The described embodiments ought to be construed to be merely  
22 illustrative of some of the more prominent features and  
23 applications of the invention. Other beneficial results can be  
24 realized by applying the disclosed invention in a different  
25 manner or modifying the invention in ways known to those familiar  
26 with the art. Thus, it should be understood that the embodiments

1 has been provided as an example and not as a limitation. The  
2 scope of the invention is defined by the appended claims.